Page No.
Date

SQL
Structured Query Language

Data : is a raw fact which describes attributes of an entity

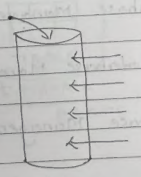
Attributes - Properties
Entity - Object

Entity - is nothing but an object it maybe Either living thing or a non-living thing
- Anything which has physical existence in a real world is known as enti

DataBase - is a place medium or a Contains used to store data in a Systematic & organized manner
- In database we can perform Several operations.
- The operations for Universely Called as CRUD operations.

4 tasks / operation

CRUD



Page No.
Date / /

SQL
Structured Query Language

Data : is a raw fact which describes attributes of an Entity

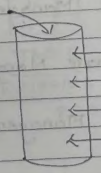
Attributes - Properties
Entity - Object

Entity - is nothing but an object it may Either living thing or a non-living thing.
- Anything which has physical existe in a real world is known as En

DataBase - is a place medium or a Contai used to Store data in a System & Organized manner
- In Database We Can perform Re operations.
- The operations for Universely C as CRUD operations.

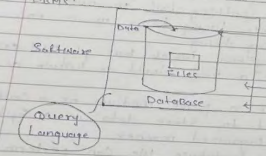
4 tasks / operation

CRUD



* DBMS (DataBase Management System):- it is a Software which is used to maintain and manages the Software Database
- DBMS has two important features
  Security and Authorization
- In DBMS we can store Data in the form of Files.
- We are using Query language to Communicate with DBMS.

DBMS:



Types of DBMS:-

i) Hierarchial Database Management System [HDBMS]

ii) Network Database Management System [NDBMS]

iii) Object oriented Database Management System [OODBMS]

iv) Relational Database Management System [RDBMS]

* Relational Model is was designed by Data Scientist E.F.CODD
- Any DBMS which supports Relational model there will becomes RDBMS

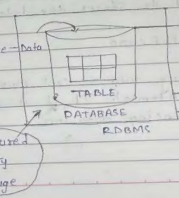
[DBMS] + [Relational Model] → [RDBMS]

- Any DBMS which follows the rules of E.F.CODD is known as RDBMS

[DBMS] + [Rules of E.F.CODD] → [RDBMS]

* RDBMS [Relational DataBase Management System]
- RDBMS is a type of DBMS Software which is used to maintain & manages the database
- In RDBMS we can store data in the form of tables. [Rows & Columns].
- We are using Structured Query Language to Communicate with RDBMS

RDBMS:

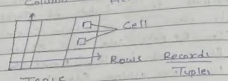


* Difference b/w DBMS & RDBMS

DBMS	RDBMS
- DataBase Management System	- Relational DBMS
- We can store Data in the form of Files	- We can store Data in the form of tables
- We are using Query lang to Communicate with DBMS	- We are using Structured Query lang to Communicate with RDBMS.

* Table:- is a logical organization of Rows & Columns and Cells.

Table:



* Cell :- is a intersection between rows & Columns
  - Cell is smallest block or a unit which is used to stores the data in a table.

* Rules of E.F.CODD

i) Data Entering into a Cell must be a Single Value data (Atomic). If it is multivalued We are facing data loss.

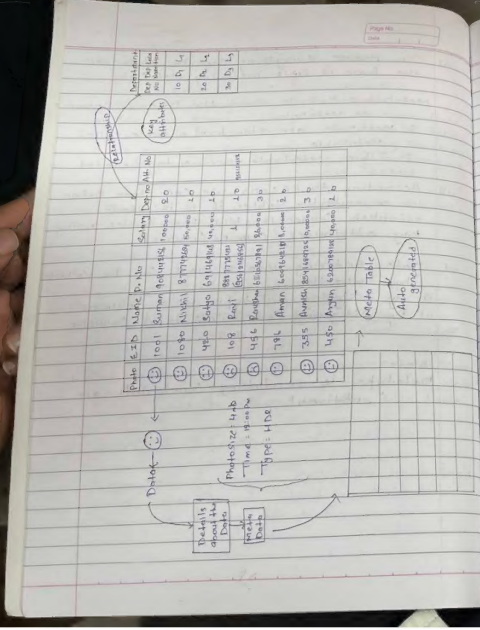
ii) In RDBMS everything can be stored in the form of table including MetaData.

MetaData :- Details about the Data is known as MetaData.
:- MetaData are stored in a table called MetaTable
:- This Metatable are autogenerated.

iii) According to EF CODD We can store Datas in a multiple tables. if it is needed establish Connection b/w the tables using key attributes (Common Column).

iv) Data Entering into a table can be Validated by two-steps
i) By assigning datatypes to a columns
ii) By assigning Constraints to a Columns.

Note :-
Datatype is mandatory Constraints are optional.



Page No.
Date

* Datatypes:- It is used to describe which kind or type of data should be entering into a particular column or a memory allocation.

- Types of Datatypes:-

(i) Char (ii) VARCHAR/VARCHAR2 (iii) DATE
(iv) NUMBER (v) LARGE OBJECT
→ Character Large object
→ Binary Large object

(i) Char :-

- 'A-Z', 'a-z', '0-9', Special character, Alphanumeric (@, !, $, #, _)
- Characters are must enclosed with the single quotations (' ').
- Syntax
  Char(size)
- The max. size of char data type is 2000 characters
- It follows fixed length of memory allocations
  e.g.: 'KRISHNA' Char(10)

K	R	I	S	H	N	A			

Used Memory → Unused Memory → Fixed length of memory Allocation

- In char datatype unused memory will be wastage of memory.
- SQL is not a Case Sensitive language But Characters datas are Case Sensitive.

(ii) VARCHAR2

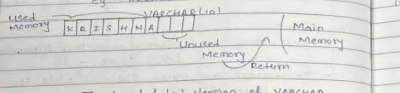
- 'A-Z', 'a-z', '0-9', special char. 'Alphanumeric' (@, !, $, #, -)
- Characters are must Enclosed with the Single quotation (' ').
- Syntax

  Varchar(Size)

The max Size of Varchar data type is 2000.

- It follows Variable length of memory allocation

  Eg: 'KRISHNA'



VARCHAR2 :- It is updated version of VARCHAR

- Syntax

  Varchar2(Size)

In VARCHAR2 we can pass 4000 characters as a max size

* Difference b/w Char & VARCHAR2

Char	VARCHAR2
- Char(size)	VARCHAR (size) VARCHAR2(size)
- 2000 ch max size	4000 ch max size
- Fixed length	Variable length

(iii) DATE:- It is used to store only the Dates

Syntax: Date

- We can store Dates in a oracle specified format

Oracle Format	
- 'DD-MON-YY'	- 'DD-MON-YYYY'
'01-JAN-25'	'31-DEC-2025'

(iv) NUMBER:- It is used to store only the numeric Values

Syntax: Number (Precision, [Scale]).

- In Number Datatype we can use 2 args
  i) Precision
  ii) Scale
- Precision is mandatory & Scale is optional.

Precision:- It is used to allocate no of digits required to store an int Values.

- The Range of precision is 1 to 38 Digits.

Scale:- It is used to store decimal Values within the given precision.

- The range of Scales is -84 to 127.
- The default Value of Scale is 0.

Eg-

Number(6): ± 999999

Number(5,3): ± 99.999

Number(3,3): ± .999

Number(3,5): ± .00999

Number(8,10): ± .0099999999

(v) Large objects:-

↳ Char Large object:- It is used to store huge amount of characters upto 4gb of size

Syntax: CLOB

↳ Binary Large object:- It is used to store the Binary Values of photo, Video, audio, documents etc upto 4GB of size

Syntax: BLOB

* Constraints :- are the rules or a Condition which are assigned to the Column for Validation

Types of Constraints :-

1) Unique
2) Not Null
3) Check
4) Primary key
5) Foreign Key

1) Unique :- It is a Constraints it Cannot Accept duplicate or repeated Values

2) Not NULL :- It is a Constraints it Cannot Accept any Null Values

Null :- It is nothing But Empty Cell
- It Cannot occupies any memory space
- 0 and space Cannot be a Null
- If We perform any Arithmetic operation With Null it returns Null Value
  e.g- 100 + Null → Null

3) Check :- It is a Constraints which gives extra Validation With the help of Conditions. If the Conditions get satisfied the Value or data get Enters into a column or else it rejects the data

e.g- check (Salary >= 0)
  5000 > 0
- Check (Length (Phone no) = 10)
  ↳ 9605492021
  10 = 10

Note
Length() : It is used to Count no. of digits or Characters.

4) Primary Key :- It is used to identify the unique records from the table.

Characteristics of primary key :-
→ We have only one primary Key in a table
→ Primary key Cannot accept duplicate or repeated Values.
→ Primary key Cannot accept null Values.
→ Primary key is a Combination of Unique & NOT Null Constraints

- Primary key is not mandatory But it is highly recommended to used the table.

c) Foreign Key:- It is used to Establish Connection b/w the tables.

Characteristics of foreign key:-

- We have n numbers of foreign keys in a table.
- Foreign key will accept duplicate or repeated Values
- Foreign key Can accept Null Values
- Foreign key is not a Combination of Unique and Not Null Constraints
- Foreign attributes to become a foreign key it must be primary key in its own table.
- foreign key presenting child table But it actually belongs to parent table.
- Foreign key is also known as referential Integrity Constraints.

* Primary Key	Foreign Key
- It Contains unique Value	- It Contains duplicate Value
- It Cannot Contain null Value	- It Contain null value
- The table has only 1 primary key	- There Can be n no. of foreign key
- used to identify the unique records from the table	- used to establish Connection b/w the table.

Employee

Emp No	Name	Salary	Dept No	Code
Table type Integer	Varchar	Number (7,2)	Number (2)	Number (4)
Constraints Primary key	Not null	Not null Default Check (salary > 0)	Not null Foreign key	Not null Check (Code in (1,2,3))
101	ABHI	15,000	10	101
102	RAJU	8,000	10	102
103	RAM	20,000	20	104
104	ABHI	12,000	20	101
110	SURYA	10,000	10	101

Child Table

Departments

Dept No	Dname	Loc
Table type Integer	Varchar2 (20)	Varchar2 (20)
Constraints Primary key	Not null	Not null
10	D_1	L_1
20	D_2	L_2

Parent table

Relations

Employee	Dept
Common Attribute	Code
common Datatype	Integer
Primary key	Yes
Not null	Yes
C_1	A_1
C_2	A_2

* Difference b/w Primary key & foreign key

PRIMARY KEY	FOREIGN KEY
-> A primary key is used to ensure data in the specific Column is unique	-> A foreign key is a column or a group of columns in a relational database table that provide link b/w data in 2 tables
-> It uniquely identifies a row in relational database table	-> It refers to the field in a table which is the primary key of another table
-> only one primary key is allowed in a table	-> More than one foreign key is allowed in a table
-> It is a Combination of UNIQUE and NotNull Constraints	-> It can Contain duplicate Values and a table in a relational database.
-> It does not allow NULL value	-> It can also Contain NULL Value
-> Its Value Cannot be deleted from the parent table	-> Its Value Can be deleted from the Child table.

* Difference b/w unique and primary

Unique	Primary
- Cannot accept Duplicate	- Cannot Accept Duplicate
- No Combination	- Combination of Unique + Null
- We can identify but they Cannot be duplicate	- It Can be identify only item
-> Does not Allows Null Value	- Does not Allow Null Value
-> Can have multiple unique constraint in a table	- only 1 primary key per table
-> Ensure Values in a Column are unique	- used to identify the unique records from table.

* overview of Sql statement

-> Data Definition language (DDL)
-> Data Manipulation language (DML)
-> Transaction Control language (TCL)
-> Data Control language (DCL)
-> Data Query Language (DQL)

-> Data Query Language (DQL)
It is used to retreive or fetch Data from the DataBase

DQL Statements:-
- Select - Display
- Projection - only Column
- Selection - Both Row & Column
- Joins - Data From multiple table

Select:- It is used to retrive data from the table and display it.

Projection:- It is used to retrive Data from the table by Selecting only the Columns.
- In projection if the Columns are get Selected by default the Values present in that Column also get Selected.

Syntax:-

SELECT */[DISTINCT] Column_name / Expression [ALIAS]
FROM Table_Name;

order of Execution:
1 -> FROM
2 -> SELECT

Selection:- It is used to retrive datas from the table by selecting both row as well as Column.

Joins:- It is used to retrive datas from the multiple table simultaneously.

* Write a query to display all the student names present in the student table.

Select Sname
From student;

student

Sno:=110	Sname = Sana	Sbanch = Civil	Sscore = 8
S.No	Sname	Branch	Percentage
1	A	MECH	90
2	B	CIVIL	85
3	C	AERO	67
4	D	ECE	65
5	E	CSE	50
6	F	IT	35

Under execution

output of From clause

Sname
A
B
C
D
E
F

Result table (or) output table

output of Select clause

Database

order of Execution:-
1 -> FROM
2 -> SELECT

NOTES:-

1. In projection, From clause will execute first.
2. In From clause we can pass table name as an arguments.
3. The job of a From clause is goes to the database & makes the table under execution.
4. After the execution of From clause, Select Clause will execute.
5. In a Select clause, we can pass 3 arguments
   1) * Astrick
   2) Column name
   3) Expressions
6. A job of a Select clause is goes to an under execution table ~~and~~ search for the column name we have mention.
7. The Select clause is used to fetch the datas from the table at display it.
8. Therefore, the Select clause is responsible for Result table (or) output table.

Commands:-

1) To set page allignments
-> Set lines 100 pages 100;

2) To clear Screen
-> cl scr (or) Clear Screen

3) To describe the table structure
-> desc table_name;
   desc student;

* Write a query to display the details from the student.

```
select *
from dept;
```

* Write a query to display student ID and percentage branch fee of the student

```
select * sid, branch, fee
from student;
```

* Write a query to display student ID and percentage, branch fee all the students

```
select * sid, branch, fee
from student
```

* Write a query to display student ID, student name, branch & percentage fee all the student

```
Select * sid, sname, branch, fee
from student;
```

Astrick * :-

It is used to display all the details from table. Along with astrick we can't use any other column name or expressions

* Write a query to display all the details present in the student table

```
select *
from student;
```

* Write a query to display all the details from the employee table

```
select *
from emp;
```

* Write a query to display all the details from the department table

```
select *
from dept;
```

- Column names

Employee table (Emp)	Department table (Dept)
EMP NO	DEPT NO
E NAME	DNAME
JOB	LOC
MGR	
HIREDATE	
SAL	
COMM	
DEPTNO	

1) Write a query to display all the details from the emp

```
Select *
from emp;
```

[illegible] the [illegible] of [illegible]

```
[illegible]
[illegible] from
from [illegible];
```

[illegible] WAQTD [illegible] the [illegible]

```
[illegible] ename, [illegible]
from [illegible];
```

4) WAQTD [illegible] the [illegible]

```
[illegible] distinct [illegible]
from [illegible];
```

5) WAQTD employee No & Department number of all the employees in emp table.

```
Select empno, deptno
from emp;
```

6) WAQTD Ename and hiredate of all the employees.

```
Select ename, hiredate
from emp;
```

7) WAQTD name and designation of all the employees

```
Select ename, job
from emp;
```

8) WAQTD name, job & Salary given all the employees

```
Select name, job, sal
from emp;
```

9) WAQTD Dnames present in different table

```
Select dname
from dept;
```

10) WAQTD Dname [illegible] different table [illegible] department

```
from dept;
```

* [illegible] it is used to remove duplicate or repeated values from the [illegible]
- We can pass distinct as a first argument to the select clause. We cannot use [illegible] as a second argument in the select clause.
- To distinct we can use multiple column. It will remove duplicate or repeated values from the combination of the columns.

11) WAQ different student [illegible]

```
Select Distinct Sname
From Student;
```

Student:

Sno	Sname	Branch	Per
1	A	AERO	70
2	B	CIVIL	80
3	C	MECH	75
4	D	ECE	75
5	A	EEE	70
6	E	CSE	65
7	B	CIVIL	80
8	F	BCOM	90

[illegible]
A
B
[illegible]
C
D
A
E
B
F

12) WAQ different Sname Branch [illegible] table.

```
Select Distinct Sname, Branch
From student;
```

Sno	Sname	Branch	Per	Sname	Branch	Sname	Branch
1	A	AERO	70	A	AERO	A	AERO
2	B	CIVIL	80	B	CIVIL	B	CIVIL
3	C	MECH	75	C	MECH	C	MECH
4	D	ECE	90	D	ECE		
5	A	EEE	70	A	EEE	A	
6	E	CSE	65	E	CSE	C	
7	B	CIVIL	80	B	CIVIL		ECOM
8	F	ECOM	92	F	ECOM		

* WAQ Different Student name Branch that percentage from student table

Sno	Sname	Branch	Per	Sname	Branch	Per	Sname	Branch	Per
1	A	AERO	70	A	AERO	70	A	AERO	70
2	B	CIVIL	80	B	CIVIL	80	B	CIVIL	80
3	C	MECH	75	C	MECH	75	C	MECH	75
4	D	ECE	90	D	ECE	90	D	ECE	90
5	A	EEE	70	A	EEE	70	A	EEE	70
6	E	CSE	65	E	CSE	65	E	CSE	65
7	B	CIVIL	80	A	CIVIL	80			
8	F	ECOM	92	F	ECOM	92			

* WAQT Different Branch, Per for all Student
  Select Distinct Sno, Branch, Per
  From student;

→ Select Sname, distinct Per
  From student; —— Error

Note: We cannot use Distinct as a second argument in the Select clause

* WAQT [illegible] Select [illegible]
  From Student;

* WAQT [illegible] Sno [illegible] Select Distinct Sno, Name
  From Student.

* WAQT Job, Salary and Hiredate [illegible]
  Select Distinct Job, Sal, Hiredate
  From Emp.

* Expression
  Any statement which [illegible] Expression
  Ex
  E + 5 = 10 → Return
  Expression
  Expression is always the combination of operands and operators
  ex: 4 + 10
  operands operator
  operands: may be a [illegible]
  operators: are the predefined symbols used to perform specific task

* WAQT annual Salary for all the employees
  Select Sal*12
  From emp;

* WAQTO employee name & total annual salary for employees.

```
Select ename, sal*12
From emp;
```

* WAQTO salary & salary hike for all emp

```
Select sal, sal+500
From emp;
```

* WAQTO name, Job & Salary with deduction of 500 for employee

```
Select name, Job, Sal-500
From emp;
```

* WAQTO Salary with hike of 10% for all employee

```
Select sal+sal*.10
From emp;
```

Note: % with hike (or) Deduction

$Sal + Sal \times \frac{K}{100}$

$Sal - Sal \times \frac{K}{100}$

* WAQTO annual Commision & annual Salary deduction of 35% for all emp.

```
Select comm*12, Sal*12-(Sal*12*35/100)
From emp;
```

↳ $Sal \times 12 + Sal \times 12 \times \frac{16}{100}$

↳ $Sal \times 12 - Sal \times 12 \times \frac{16}{100}$

* Enter 2 or 3 employee Annual Salary for all employee

Note: If we want to fetch all the rows along with column name or expression we use tablename.*

* [ALIAS]: it is a alternative name given to the Columnname or expression in the result table.

→ We can give alias with or without using keyword.

→ As keyword

→ Alias will be a single string

→ If it is a multiple string then use double quotation (" ") or (Inverted ' ').

```
Select Sal*12 As "Annual Salary"
From emp;
```

Eg:

```
Sal*12 Annual_Salary
Sal*12 "Annual Salary"
Sal*12 As "Annual Salary"
Sal*12 As "Annual_Salary"
```

* WAQTO Annual Salary & Annual Commision with deduction of 100Rs for All the employees

```
Select Sal*12 As "Annual sal", Comm*12-100
As "Annual Comm Deduction"
From emp;
```

[illegible]

[illegible]

Full Form.

- WAQTP Name of the employee along with their Annual Sal.
  Select name, Sal*12 As "Annual Sal"
  from emp;

2. WAQTP Name and Job Job of the employee along with their Annual Salary.
  Select ename, Sal*12 As "Annual Salary"
  from emp;

3. WAQTP All the details of the employee along with an Annual Bonus of 500.
  Select empno, Sal*12+500 as "Annual Salary"
  from emp;

4. WAQTP Name, Salary and Salary with hike of 10%
  Select ename, Sal+Sal*10/100 as "Hike"
  from emp;

[illegible]

[illegible]

[illegible]

- Where is used to filter the records
- Where is used along the from clause
- Where is used to filter the rows
- Where is used in update statements
- Where is used in delete statements
- Where is used in select statements
- For Numeric Values / Date Values / Text
- Remember text and values / Text
- Always Single Quote We can write multiple
- Where clause will use lots of logical operators

* WAQTD Name & Deptno of employees if the employees are working in Deptno 10

```
Select Ename, Deptno
From emp
Where Deptno=10;
```

Output of From Clause

Empno.	Ename.	Job.	Deptno.
1	SCOTT	1500	30
2	JAMES	2500	10
3	ALLEN	3500	20
4	MILLER	4500	10
5	WARD	5500	30
6	SMITH	6500	20
7	KING	7500	10

Output of Where Clause

	Empno	Ename	Job	Deptno
1	4	MILLER	1950	10
2	7	KING	4000	10

Output of Select Clause

Ename	Deptno
JAMES	10
MILLER	10
KING	10

Under Execution

After Condition Evaluation

* WAQTD Name & Salary of the employees if employee are getting Salary less than 2000

```
select ename, sal
from emp
where sal<2000;
```

* WAQTD Details of an employees Where Working as a manager

```
Select *
From emp
Where Job='MANAGER'
```

* WAQTD Name, Job & HireDate of employees if employee Are Hired After 31st Dec 82

```
Select ename, Job, Hiredate
From emp
Where Hiredate > '31-DEC-82'
```

* WAQTD Details of employee along with annual sal if annual sal is less than 25000

```
Select emp.*, Sal*12 As "Annual Sal"
From emp
Where Sal*12 < 25000;
```

Note:- We cannot use ALIAS Name in Where Clause. Because Where Clause Executes Before Select Clause.

Assignment

1. WAQTD Details of employee whose name is ...

```
Select *
From emp
Where ename ...
```

2. WAQTD NAME of emp whose job is ...

```
Select Ename, Sal
From emp
Where Job ...
```

3. WAQTD SALARY of Emp whose job is SALESMAN

```
Select Sal, Job
From emp
Where Job = 'SALESMAN';
```

4. WAQTD Details of Emp who earns more than 2000

```
Select *
From emp
Where Sal > 2000;
```

5. WAQTD Details of Emp who was Hired After 01-JAN-81

```
Select *
From emp
Where hiredate > '01-JAN-81';
```

6. WAQTD Details of emp whos name is 'JAMES'

```
Select *
From emp
Where ename 'JAMES';
```

* WAQTD Details of employee whose name is ...

```
Select emp
From emp
Where ename = 'MILLER';
```

* WAQTD Details of employee & salary with hike of 15% in the salary for SALESMAN

```
Select emp*, SAL + SAL*15/100 As "SAL HIKE"
From emp
Where Job 'SALES';
```

[illegible] NAME and SAL Along [illegible] Salary is more than 10000

[illegible] the ANNUAL Salary is more than 10000

Select ename, sal, sal*12 as Annual_Sal

from emp

where sal*12 > 10000;

8 | WRITE ENAME of the employee who are [illegible] DEPT 3

Select ename

from emp

where deptno = 3;

9 WRITE ENAME and HIREDATE of they are hired [illegible] 1981

Select ename, hiredate

from emp

where hiredate < '01-JAN-81';

10 WRITE Details of EMP identify A MANAGER

Select *

from emp

where job 'MANAGER';

11 WRITE NAME and SALARY of Emp Earns A [illegible] of 5000

Select ename, sal

from emp

where [illegible];

12 WRITE DETAILS of Emp Having Commission More than [illegible]

Select *

from emp

~~where hiredate < '01-JAN-~~

comm > 500;

13 WRITE ENAME of Employee who [illegible]

[illegible]

from emp

[illegible]

14 WRITE Details of employee who [illegible]

Select *

from emp

where sal 'ANALYST';

15 WRITE Details of Emp Having more than [illegible]

Select *

from emp

where sal > 5000;

2. [illegible]

i. Arithmetic operators :- (+ , - , * , /)

ii. Concatenation operator :- (||)

iii. Comparison operator :- (= , ! = , < , > , <= , >=)

iv. Relational operator :- (= , < , > , <= , >=)

v. Logical operator :- (AND, OR, NOT)

vi. Special operator :- ((,) , IN

NOT IN

BETWEEN

NOT BETWEEN

IS

IS NOT

LIKE

NOT LIKE

ALL

ANY

EXISTS

NOT EXISTS

[illegible]

The given string:

The symbol for concatenation operator is

'[illegible] MR ' || ENAME

From EMP;

MR ' || ENA

[illegible]

[illegible]

|| 'SMITH HR SALARY IS [illegible]

Select 'MR' || ENAME || 'HR' SALARY IS' || SAL

From EMP;

[illegible]

Select '[illegible] || ENAME || 'OK OF NAME IS SMIT[illegible]

FROM EMP

WHERE ENAME = 'SMITH';

[illegible] It is used to [illegible] multiple

Conditions in a where clause.

AND, OR, NOT.

[illegible] It is a Binary Multiplication

operator

- AND operator return true if all the

conditions get satisfied

[illegible]

[illegible]

[illegible] it is a binary [illegible]

[illegible] return true if [illegible]

get satisfied

[illegible]

[illegible] It is a binary operator [illegible]

also known as inverse operator.

- NOT operator is similar to Unary [illegible]

It is used to negate the [illegible]

Define [illegible]

Not Define [illegible]

v. WANTD Details of [illegible]

Select *

from emp

where Job = 'MANAGER' AND DEPT[illegible]

8 WAD [illegible]

Sel[illegible]

Select [illegible], Job, [illegible]

From emp;

where Job = 'Salesman' ~~[illegible]~~ AND Sal => 1000 AND DeptNo = 3

select *
from emp
where Job = 'analyst' or Job = 'clerk';

select *
from emp
where Job = 'salesman' or Job = 'clerk';

select *
from emp
where Deptno = 10 and Deptno = 20;

select *
from emp
where Not(Job='MANAGER' or Job = ...

select *
from emp
where Job='clerk' AND earning < 1500

Select ename, Deptno, ...
From emp
Where Job = 'clerk' AND (Deptno=10 ...

Select *
From emp
Where (Job='clerk' ...

Select ename
From emp
Where Job, Deptno in ...

[illegible]

[illegible]

Select *
from emp
Where Sal Not Between 1250 and 1500;

Select ename, Sal
from emp
Where Sal not between 1000 and 1500;

Select hiredate
from emp
where hiredate not between '01-JAN-81' and '31-DEC-81';

Select *
from emp
where Sal Between 1250 and 1500;

Select *
from emp
where hiredate between '01-JAN-81' AND '31-DEC-81';

Select ename, Sal
from emp
where Sal Between 1000 and 1500;

Select *
from emp
where hiredate Between '01-JAN-81' And '31-DEC-81';

Select *
from emp
where hiredate between '01-JAN-81' and '31-DEC-81';

It is similar to less or greater than
it checking for range in numeric Values.

Syntax
Column-name/expression NOT BETWEEN

[illegible] is used to compare with only the [illegible]

[illegible] comm or is null;

[illegible]

```
select ename,sal
from emp
where sal is null;
```

[illegible]

```
select *
from emp
where mgr is null;
```

[illegible] To be used to compare with not null values

[illegible] function/expression IS NOT NULL;

[illegible]

```
select *
from emp
where comm is not null and comm != 0;
```

[illegible]

```
select *
from emp
where sal is not null and comm is null;
```

[illegible] To be used to compare two values.

[illegible] expression [illegible]

[illegible] the pattern is case sensitive. We use the special characters

[illegible]

[illegible] To accept a number or character any character for a character or numbers

[illegible] The one underscore for accept exactly only one character but any type [illegible]

[illegible]

```
select ename
from emp
where ename like 'M%';
```

[illegible]

```
select *
from emp
where ename like '%E';
```

[illegible]

```
select ename
from emp
where ename like '%%' and '%%';
```

```
select ename
from emp
where ename not like '%C%';
```

```
select count(*)
from emp
where hiredate like '%-DEC-%';
```

```
select *
from emp
where ename like '[illegible]';
```

```
select ename
from emp
where ename like '%A%'
      and '%A%';
```

```
select ename,sal
from emp
where ename like '%[illegible]';
```

```
select ename
from emp
where ename not like '%[illegible]';
```

```
select ename
from emp
where ename not like '%[illegible]';
```

```
select ename
from emp
where ename not like '%[illegible]';
```

```
select hiredate
from emp
where hiredate not like '[illegible]%'
      '%[illegible]';
```

```
select ename
from emp
where ename Notlike '%[illegible]%';
```

```
select ename
from emp
where ename like '%A%' and ename not like '%A%%'
```

```
select ename
from emp
where ename not '%A%A%'
```

```
select ename
from emp
where ename like '%B%' and ename not like '%B%%'
```

```
select *
from emp
where ename like A%';
```

```
select *
from emp
where ename like '%';
```

```
select *
from emp
where ename like 'A%%';
```

```
select *
from emp
where ename like '%';
```

```
select *
from emp
where ename like '%A%';
```

```
select *
from emp
where ename like '%L%';
```

```
select *
from emp
where ename like '%L%' or '%L%';
```

```
select *
from emp
where ename like 'A%M%' or 'A%M%';
```

```
select *
from emp
where ename like '____';
```

[illegible]

[illegible] hire in month of Feb

Select ename, hiredate

from emp

Where hiredate like '%Feb%';

[illegible]

Select ename, sal

from emp

Where sal like [illegible]

[illegible]

Select ename, hiredate, sal

from emp

Where [illegible] like '[illegible]' AND hiredate like '%[illegible]';

[illegible]

Select ename, sal

from emp

Where sal like [illegible] AND ename like '%[illegible]';

[illegible]

Select ename

from emp

Where ename like '[illegible]';

[illegible]

from emp

Where Deptno [illegible]

3. WRITE name [illegible]

Select [illegible]

from emp

Where hiredate [illegible]

[illegible] WRITE Name [illegible] Salary [illegible]

Select ename, sal

from emp

Where Comm. is Not Null;

[illegible] WRITE Details of emp Working [illegible]

Having character 'A' [illegible]

Select *

from emp

Where Job = 'clerk' and Deptno In (10, 30) [illegible]

[illegible] WRITE names of emp having char [illegible]

Select ename

from emp

Where ename like '%s';

[illegible] name, hiredate, of emp hired after [illegible]

Select ename, hiredate

from emp

Where hiredate between [illegible]

7. WAQTD [illegible] of employees [illegible] as analyst & [illegible]

```
Select *
From emp
Where Job 'Analyst' and Sal [illegible]
```

8. WAQ [illegible] names of the employees [illegible]

```
Select ename
From emp
Where Hiredate Like '%7%',
```

9. WAQTD Details of the employees Working As Salesman And [illegible]

```
Select *
From emp
Where Job 'Salesman' And [illegible]
```

10. WAQTD Name and Salary Given to the emp Hired during 1989 in a Dept in [illegible]

```
Select ename, Sal
From emp
Where hire date between '01-Jan-1989' and '31-Dec-1989'
Deptno [illegible] In (10,40);
```

11. WAQTD Details of emp Having L in Their name and Works as Manager

```
Select *
From emp
Where ename Like '%L%' and Job 'Manager'.
```

12. WAQ [illegible] names [illegible]

```
Select ename
From emp
Where [illegible]
```

13. WAQ [illegible] of the emp [illegible]

```
Select ename
From emp
Where ename Like '[illegible]'
```

14. List All the Emp Names [illegible] Name has 'A' As the [illegible]

```
Select ename
from emp
Where Not Like '%A%'
```

15. List the Details of the emp Working [illegible] and Hired After 1984 and has A Name [illegible] with [illegible]

```
Select *
from emp
Where Job='Manager' AND Hiredate [illegible]
AND ename Like '%[illegible]'
```

- Function - the set of rules to be followed or a step used to perform specific task

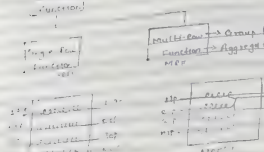
Functions

- Built-Function

User Defined function

- Single Row Function

Multi-Row Function → Group function
→ Aggregate fun
MRF



- Ex:- If Executes Row by Row
- If We pass one input it single Row function
- If Passes execute & generates one output.
- If We pass n no. of inputs in Single Row Function it generates n no of outputs.

Emp

Ename	Sal
ALLEN	1600
MILLER	1300
Tom	1950
JAMES	4700
[illegible]	[illegible]

Length:- To count the no of character

Select Ename, Length (Ename)
From Emp;

- Output
- Ename
- Length
- ALLEN 5
- MILLER 6
- TOM 3
- JAMES 5
- [illegible]

- Group Function / Aggregate Function
- It is Row Fun Aggregates or the Row and Group Select and Execute and then gives one output
- If We pass one input it gives one output
- If We pass n no of inputs it generates one output

List of Multi-Row-Function

- Max()
- Min()
- Sum()
- Avg()
- Count()

- WAQTO max sal present in emp table

Select Max(Sal)
From emp;

- output
- Max(sal)
- 4700

Rules of Multi Row Function

- In multi row fun we can use single arg/input. The argument is may be col name or expression. Along with them, we cannot use arg/colname + expression.
- Max() ignores the null values.
- The where use multiple row i.e. in where clause.
- Count is the only multiple row function which accepts * as a argument.

```
select max(sal) ,min(sal)
from emp.

select * count(*)
from emp
where job 'salesman'
```

- [illegible] are hired before the [illegible]

```
select count(sal)
from emp.
where hiredate ~~between '01-JAN-80' and '31-DEC-80'~~
and comm is not null;
```

- [illegible] is null [illegible] is [illegible]

```
select count(*), count(sal)
from emp
where ton in('salesman','analyst') And Deptno=30 and sal;
```

```
select count(*)
from emp
where sal > avg(sal)
```

- Avg() accepts the [illegible] behaviour given to a expr in [illegible] and to treat it as a normal function.

```
select colname/expression like/not like 'Pattern to match' from emp
```

Note:
1 Escape character must be defined
2 Escape char must be used before the special character (% or _) which has to treated as a normal character.
3 The recommended character is [!, /, \, #].

Table

110	[illegible]	* WHAT'S Name [illegible]
150	Dept-name	
100	Colleague	
302	Regn-Mgr	
[illegible]	[illegible]	

```
select Ename
from emp
where Ename like '[illegible]'
```

```
Select Count(*)
From emp
Where Job = 'Clerk';
```

```
Select Avg(Sal), Sum(Sal), Count(*), Max(Sal)
From emp
Where Job = 'President';
```

select count(*)
from emp
where Ename Like '%a%';

select count(distinct sal)
from emp
where ename like 'a%';

select count(distinct mgr)
from emp;

select count(*)
from emp
where Ename Like '%a%';

select count(*)
from emp
where Job = 'clerk';

select count(*)
from emp
where Job In('Manager', 'clerk') and Deptno
= 30;

select count(*)
from emp
where Job In('Manager', 'clerk') and Deptno
In(10, 30);

select count(*)
from emp
where Job = 'clerk';

select avg(sal)
from emp
where Job In('Manager', 'clerk') and Deptno
In(10, 30);

select count(*)
from emp
where Job = 'clerk' and Deptno In(10, 30);

[illegible] used to group the records

GROUP BY [illegible] expression (one or more)
[illegible] table_name
[illegible] (condition)
[illegible] column name (expression)

Order of execution
- FROM
- WHERE (if used) [Select by Row]
- GROUP BY [Select by Group]
- SELECT [Select Column]

- GROUP BY clause is used to group the records
- The Executor runs by row
- The group by clause we can use Column/Expression as an argument
- After the execution of group by clause we can perform groups
- Therefore any clause Executes after the group by
  - HAVING Executes group by group
- Without Where clause we can use group by clause

Select Count(*), Deptno from [illegible] group by
[illegible]
Group by Deptno;

ALLEN
JAMES
MILLER
BLAKE
FORD

count(*)

```
select count(*), Job
from emp
group by Job;
```

```
select count(*), min(Sal), deptno
from emp
where Job in ('manager', 'salesman')
group by deptno;
```

```
select count(*), Deptno
from emp
where Deptno in 30
group by Deptno;
```

```
select Avg(Sal), Job
from emp
where Group by Job;
```

```
select count(*), Job
from emp
where Ename like 'M%'
group by Job;
```

```
select count(*)
from emp
where Job in ('president', 'manager');
group by deptno;
```

```
select count(*)
from emp
where Job = 'manager'
group by deptno;
```

```
select count(*)
from emp
where Sal > 2000
group by deptno;
```

```
select count(*), Avg(Sal)
from emp
where Sal > 2000
group by deptno;
```

```
select job, sum(salary)
from emp
where job 'salesman'
group by deptno;

select max(salary), min(salary)
from emp
group by job;

select deptno, job, sum(salary), avg(salary)
from emp
where sal > 1000
group by deptno, job
order by deptno, job;
```

```
select count(*), sum(sal)
from emp
group by deptno
order by deptno;
```

```
SELECT group_by_expression / aggregate_function
FROM Table_name
[WHERE [filter_condition]]
GROUP BY Grouping_expression
HAVING [group_filter_condition]
```

```
SELECT [DISTINCT]
FROM
WHERE        [row filter]
GROUP BY     [group by]
HAVING       [group filter]
ORDER BY     [order by]
```

```
Emp
EMPNO  Ename  Job  Mgr  Hiredate  Sal  Comm  Deptno
1  ALLEN  1980  SL  7  1600  300  30
2  BLAKE  1980  MG  7  2850  30
3  JAMES  1980  10  T  950  30
4  MILLER  1980  20  7  1300  10
5  FORD  1980  SL  7  3000  20
6  ADAMS  1980  MG  7  1100  20
7  TURNER  1980  10  7  1500  0  30
8  KING  1980  10  7  5000  10
UNDER EXEC  1980  7  1000  30
```

```
4  MILLER  1980  SL
6  ADAMS  1980  MG
8  KING  1980  MG
```

Note:

- Having clause is used to filter the groups
- Having clause [illegible] group by group
- The Having clause we can use group filter conditions as an argument (Where can't do)

Where: group by clause we cannot use Having clause

- [illegible] no of [illegible] greater than [illegible]
  [illegible] an [illegible] of [illegible] Department

```
select count(*), Deptno  Group by [illegible]
from emp
where sal > [illegible]
Group by Deptno
Having count(*) > 3;
```

```
select count(*)
from emp
Group by Sal
Having count(*) > 1;
```

```
select count(*), Deptno
from emp
where Sal
Group by Deptno
Having count(*) > 3;
```

```
select max(sal) job
from emp
Group by Job
Having max(sal) > [illegible]
```

```
Select Sal
from emp
Group by Sal
Having Count(*) > 2;
```

```
Select : hiredate
from emp
Group by hiredate
Having Count(*) > 2;
```

```
select Deptno, Count(*) Deptno
from emp
Group by Deptno
Having Count(*) > 2;
```

```
Select Job, Count(*) Job
from emp
where sal > 1500
group by Job
Having Count(*) > 5;
```

```
Select Avg(sal)
from emp
group by deptno
Having Avg(sal) > 2000;
```

```
Select deptno
from emp
where ename like '%M%' or '%A%';
group by deptno
Having count(*) > 2;
```

```
Select Ename
from emp
where Job In ('Manager ***, 'Salesman')
order by Ename desc;
```

```
Select count(*), Job
from emp
where ename like '%K%'
Group By Job
order By Job DESC;
```

```
Select
Lower(  )
Upper(  )
Initcap(  )
Length(  )
Substr(  )
Instr(  )
Trim(  )
Replace(  )
Translate(  )
```

```
Select Ename
from emp
order by Sal desc;
```

```
Select Ename
from emp
where Job In ('Manager ***, 'Salesman')
order by Ename desc;
```

```
Select count(*), Job
from emp
where ename like '%K%'
Group By Job
order By Job DESC;
```

* WAQTO Max Sal given to Emp and Deptno attributes of emp are working in each department. Arrange the values in dep wise

```
Select Max(Sal), Deptno
from emp
Group By Deptno
Having Count(*) >=2
Order By Deptno;
```

* Subquery:- A query return inside another query is known as Subquery.

Working Flow:-

Result output (Outer Query) ← Dependent

↑ ↓

(Inner Query)

- Let us consider two query outer query & Inner query
- In Subquery inner query will execute first and generates a output.
- The output of an inner query is sending as a input to the outer query
- With the help of this input outer query will execute and generates an result and an output table.
- Therefore outer query is dependent on inner query.

* When and Why we use Subquery?

Case 1

When there is unknown condition present in a query, we use subquery.

Emp:

Ename	Sal	Deptno
ADAMS	1100	20
JONES	1950	10
BLAKE	2850	30
FORD	4000	10
KING	1250	30

Case: Unknown condition

Select Ename

* WAQTO Names of emps who are earning sal greather than 1500

```
Select Ename
From Emp
Where Sal > 1500;
```

Direct Condition or Known Condition

* WAQTO Names of emp who are earning sal greather than Jones.

```
Select Ename
from Emp
Where Sal > 'JONES';
```

Indirect Condition or Unknown Condition

```
Select Ename
From Emp
Where Sal > (Select Sal
             from emp
             Where Ename = 'JONES');
```

→	1100	F
→	1950	F
→	2850	T
→	4000	T
→	1250	F

JONES	1950	10

* WAQTD Name & Hirdate of an emp if Emps are hired before the Miller

```
Select Ename,hirdate
from emp
Where Hirdate < (Select Hirdate
                 from emp
                 Where Ename='Miller');
```

* WAQTD Names of employees who are working in the same designation of adams.

```
Select ename
from emp
Where Job=(Select Job
           from emp
           Where Ename='adams');
```

* WAQTD Details of "Emps who are working in the same dept of Jones

```
Select *
from emp
Where deptno (Select deptno
              from emp
              Where Ename='JONES');
```

* WAQTD Details of Emps who are working as a manager and hired after the Smith

```
Select *
from emp
Where Job='Manager' AND hirdate>
                (Select hirdate
                 from emp
                 Where Ename='Smith');
```

* WAQTD Name & Sal of Emps if Emps are earning Sal less than the Blake and emp name has Char A.

```
Select Ename,Sal
from emp
Where Ename Like '%A%' AND Sal < (Select Sal
                                  from emp
                                  Where Ename='BLAKE');
```

* WAQTD Details of emps who are earning Sal greater than Jones and less than King

```
Select *
from emp
Where Sal > (Select Sal                AND Sal < (Select Sal
             from emp                              from emp
             Where Ename='Jones')                  Where Ename='King');
```

* WAQTD Details of Emp who are hired after ~~adams~~ Allen but before a Adams & Emp hired year of 81.

```
Select *
from emp
Where hirdate Like '%81' AND Hirdate >(Select hirdate
                                       from emp
                                       Where Ename='ALLEN') AND Hirdate < (Select hirdate
                                                                           from emp
                                                                           Where Ename='ADAMS');
```

* WAQTD Details of Emp along with Annual Sal if Emps are working in same dept of Turner and same designation of Martin and earning Sal greater than Smith

```
Select emp.*, Sal*12
from emp
Where deptno=(Select ~~ename~~ deptno
              from EMP
```

Where Ename = 'Turner' AND Job = (Select Ename
from emp
where Ename = 'WARD')

Sal > (Select Sal
from emp
where Ename = 'SMITH');

CHEDULE

WEDNESDAY	THURSDAY	FRIDAY

my self

Name: Aman Raj

Class:

School / College:

Email:

Index

Date	Page No.	Subject	Sign	Remark